PyPES Library Manual

Marat Sibaev, Deborah Crittenden[*]

May, 2015

Contents

[*]Department of Chemistry, University of Canterbury, Christchurch, NZ

1 Introduction

For those interested in an overview of what PyPES library is about and how it works, please refer to the original publication. [1] A pre-print version is included in the distribution documentation for convenience, together with a .csv file containing our VCI results for all force-fields.

This manual contains a list of all input commands, along with a list of implemented force-fields, and a few small tutorials on how to utilise different features of the program. The tutorials are also included in the PyPES release within the tutes/ directory.

The code is released for everyones use, freely. You are welcome to use the code as you please, and incorporate parts of it into your own programs. However, we do ask that you give credit where it's due and please cite the original paper if you find this work useful in your research. If you find any bugs in our implementation, please send the input file with description of the error to deborah.crittenden@canterbury.ac.nz.

2 Setup and usage

PyPES requires the following programs and packages to be properly installed and configured:

- Python 2.7+ or Python 3.0+
- SciPy 0.13.2+
- NumPy 1.8.0+
- SymPy 0.7.4.1+
- Cython 0.21+

PyPES has been tested with Python 2.7.6 and 3.4.1, with the packages listed above. PyPES should be compatible with more recent versions of the listed programs and packages, and is also probably backward compatible within major versions. We strongly recommend using a package manager (MacPorts, HomeBrew, etc) to install the required Python and Cython distributions.

After downloading the compressed and zipped PyPES.tgz file, it should be moved to an appropriate install directory and extracted:

```
tar xvfz PyPES.tgz
```

The next step is to compile the computationally intensive modules that have been optimized and converted to C code using the Cython package. This is most easily achieved using a Cython install and the provided `setup_cyth.py` script:

```
cd PyPES/code
make_clean
python setup_cyth.py build_ext --inplace
```

The C modules may also be compiled using a standard C compiler (see Cython project website for details) but this is not recommended. Once the C modules have been compiled and the correct paths added to your system, the format for calling PyPES is:

```
pypes_lib.py input_file (output_file)
```

where formats for `input_file` and `output_file` are explained in the tutorials. Specifying the `output_file` is optional and can be ignored if you are only interested in obtaining derivative data written to separate files.

For those interested in calling PyPES from their code, as they would an *ab-initio* package, wrappers are included in the `code/wrappers` directory.

Atomic units are used within the program and all output data is in atomic units as well. Conversion constants can be found in `constants.py`.

3 Input options

Before describing the PyPES input options, we first define terminology used to describe coordinate systems implemented within PyPES (Table 1), and special characters used within the input file (Table 2).

Table 1: Summary of internal coordinates that have been implemented. See original paper for definitions. BL = bond length, BA = bend angle, DA = dihedral angle, OOP = out-of-plane coordinate, SPF = Simons-Parr-Finlan radial coordinate, CH = Carter-Handy bend angle coordinate.

Symbol	Usable name	Symbol	Usable name
r	`BL`	τ	`DA`
$f_{\mathrm{M}}(r)$	`Morse`	$\sin(\tau)$	`SinDA`
$f_{\mathrm{SPF}}(r)$	`SPF`	ω	`OOP1`
θ	`BA`	$\sin(\omega)$	`SinOOP1`
$\cos(\theta)$	`CosBA`	ω'	`OOP2`
$f_{\mathrm{CH}}(\theta)$	`CH`	$\sin(\omega')$	`SinOOP2`

Table 2: Special characters that can be used in `input_file`.

Character	Description
`#`	line is ignored, when used as a first character on that line;
`;`	can be used to wrap the line;
`STOP`	stops reading commands;
`@@@`	indicates end of job-block.

Command	Options / Default / Description

`COMMENT`	`"your comment, use semicolon ;`
	`to wrap the line"`
	Default: None
	Specified comment must be enclosed in quotation marks.

`MOLECULE`	`name` Default: None A valid name **must** be specified to determine which force-field module to load. Names of implemented force fields are given in Table 4. Multiple names separated by empty space indicate multiple jobs per input block.
`PESN`	`integer` $\geq$ `1` *Default:* `1` Specifies which force-field within the module to use. Valid potential energy surface numbers for each molecule are listed in Table 4
`DLVL_INT`	`-1 to 6` *Default:* `0` Derivative level for derivatives of PES with respect to internal coordinates, and coordinate system in which force field is defined (see tutorial 4). `-1` – no derivatives are calculated (useful if only geometry optimisation is required); `0` – energy is calculated (zeroth derivative); `1` – first derivative; etc.
`DLVL_CART`	`0 to 6` *Default:* max(`0`, `DLVL_INT`)

Continued on next page

Table 3 – *Continued from previous page*

Command	Options / Default / Description
	Derivative level for derivatives of PES with respect to Cartesian and normal mode coordinates, generated through coordinate transformation.
`PRINT_DERIV_INT`	`True or False` *Default:* `True` Prints derivatives of PES with respect to internal coordinates, to the `output_file` if given, or to separate data files if `PRINT_DERIV_FILE` is set explicitly, or both.
`PRINT_DERIV_CART`	`True or False` *Default:* `False` Prints derivatives of PES with respect to Cartesian coordinates, to the `output_file` if given, or to separate data files if `PRINT_DERIV_FILE` is set explicitly, or both.
`PRINT_DERIV_NM`	`True or False` *Default:* `True` Prints derivatives of PES with respect to normal mode coordinates, to the `output_file` if given, or to separate data files if `PRINT_DERIV_FILE` is set explicitly, or both.

`PRINT_DERIV_FILE`	`file_name`
	Default: None
	If `file_name == "default"`, then
	`file_name = "derivs_" + MOLECULE + "_" + PESN`
	Writes derivatives into separate files for each derivative level (suffixed with `_DnV` where n = derivative level). Only derivatives set using `PRINT_DERIV` commands above will be written.
	Upper triangular matrix elements ($1 \leq i \leq nvar; i \leq j \leq nvar; etc$) are printed line by line, and wrapper scripts for restoring the derivative matrices in Python, C, FORTRAN, MATLAB and Mathematica are included with the program.

Continued on next page

Table 3 – *Continued from previous page*

Command	Options / Default / Description
	To distinguish between derivatives with respect to different coordinates, extensions are added to the file names: `.d_int`, `.d_cart` and `.d_nm` for internal, Cartesian and normal mode coordinates, respectively.

`MASSES`	None
	Default: as stored in the force-field module
	Indicates that new masses are specified. New masses may be specified within the `MASSES` section in one of two ways (see also Tutorial 2)
	`MASSES`
	`atom_index   mass in a.m.u.`
	or
	`atom_index   new_atom_type`
	`END`
	Both specification formats may be used within a single `MASSES` section, and `MASSES` takes precedence over atom types assigned in `CARTESIAN`
	Available atom types within PyPES correspond to common atomic isotopes, with the isotope number specified after the element label, *e.g.* ^{1}H = H1, ^{2}H = H2, etc. Unnumbered atom labels correspond to most common isotope of each element *e.g.* ^{1}H = H1 = H.

`CARTESIAN`	`bohr or angstrom`
	Default: unit = `bohr`, equilibrium geometry loaded from force field module for `MOLECULE`
	User-specified geometries must be in the following format (see Tute 2):
	`Cartesian bohr/angstrom`
	`atom_type X Y Z`
	⋮

Continued on next page

Table 3 – *Continued from previous page*

Command	**Options / Default / Description**
	Available atom types within PyPES correspond to common atomic isotopes, with the isotope number specified after the element label, *e.g.* ^{1}H = H1, ^{2}H = H2, etc. Unnumbered atom labels correspond to most common isotope of each element *e.g.* ^{1}H = H1 = H. The order and identity of atoms must be the same as in the saved force-field (also given in Table 4), but isotopic substitutions are allowed using the isotope numbering convention described above.

`INTERNAL`	`bohr or angstrom`
	Default: unit = `bohr`, equilibrium geometry
	Indicates that new geometry in internal coordinates is specified, in the following format (see Tute 2):
	`INTERNAL bohr/angstrom`
	`i_int value`
	⋮
	`END`
	where `i_int` is internal coordinate index number and `value` its desired value.
	The geometry must be specified with respect to the $\{r,\theta,\tau,\omega$ or $\omega'\}$ coordinate system, *i.e.* even if a force-field is defined in terms of $\{f_M(r), \cos(\theta), \sin(\tau)\}$ coordinates, the specified `value` must correspond to $\{r,\theta,\tau\}$ coordinates. All angular coordinates must be specified in degrees.
	The corresponding geometry in Cartesian coordinates is obtained by minimising root-mean-square-deviation from specified internal coordinates. If `CARTESIAN` is specified concurrently, then the given Cartesian geometry is used as a starting point. Otherwise, the equilibrium geometry is used.

`OPT_GEOM_WRT_E`	`True or False` Default: `False` Optimises geometry with respect to energy, using the `minimize()` function from SciPy.
`MASSES_FILE`	`file_name`

Continued on next page

Table 3 – *Continued from previous page*

Command	**Options / Default / Description**
	Default: None File specifying masses, should contain `MASSES` section in format described above. No special characters or empty lines are allowed. Takes precedence over `MASSES` in `input_file`.
`COORDS_CART_FILE`	`file_name` Default: None File specifying geometry in Cartesian coordinates, should contain `CARTESIAN` command in format described above. No special characters or empty lines are allowed. Takes precedence over `CARTESIAN` in `input_file`.
`COORDS_INT_FILE`	`file_name` Default: None File specifying geometry in internal coordinates, should contain `INTERNAL` command in format described above. No special characters or empty lines are allowed. Takes precedence over `INTERNAL` in `input_file`.
`DO_NORMAL_MODE`	`True or False` Default: `False`, unless `PRINT_DERIV_NM` = `True` Performs normal mode analysis. Requires at least second derivatives of PES with respect to Cartesian coordinates to be calculated. Harmonic frequencies will be printed before and after projection of rotational and translational degrees of freedom, together with Cartesian vectors defining normal mode coordinates ($\frac{\partial \mathbf{X}}{\partial \mathbf{Q}}$ coefficients). **Warning:** Normal mode coordinates are only defined at equilibrium, it is not valid to perform normal mode analysis off equilibrium. If requesting normal mode derivatives at an off-equilibrium geometry **it is your responsibility** to provide valid normal mode coordinate vectors calculated at equilibrium, via the `NEW_NM_COORDS` keyword.
`NEW_NM_COORDS`	`file_name` Default: None

Continued on next page

Table 3 – *Continued from previous page*

Command	**Options / Default / Description**
	Reads in new set of normal mode coordinates, $\frac{\partial \mathbf{X}}{\partial \mathbf{Q}}$, from `file_name` and transforms derivatives of PES with respect to Cartesians to the derivatives with respect to the supplied coordinates. It is easiest to obtain $\frac{\partial \mathbf{X}}{\partial \mathbf{Q}}\|_{\mathrm{eq}}$ from a previous PyPES calculation at equilibrium, see Tute 3.
`NEW_INT_TYPE`	1a) `i_int/new_int_type` ; identify internal by its index and specify its new internal type (e.g. `2/SinDA`) 1b) `i_int/new_int_type/int_params` ; same as 1a) only add necessary internal parameters (e.g. `5/Morse/1.0`) 2) `old_int_type/new_int_type` ; change all internal coordinates of type `old_int_type` to type `new_int_type`, internal parameters cannot be specified (e.g. `Morse/SPF`) Default: None **Important note:** No spaces are allowed between type specifiers and forward slash characters, the whole argument must be one continuous string. Transforms derivatives of PES with respect to one set of internal coordinates, into a new set of internal coordinates. Valid `int_type`s are given in Table 1, `int_params` are used in defining parameterized coordinates (Morse, Carter-Handy). When transforming into Morse or Carter-Handy coordinates internal parameters can only be specified using option 1b), otherwise they will be calculated automatically, using $\alpha = \frac{\partial^3 \mathrm{V}}{\partial r^3}/(3\frac{\partial^2 \mathrm{V}}{\partial r^2})$ for Morse parameter and; $\beta_1 = 1.0$, $\beta_2 = \frac{\partial^3 \mathrm{V}}{\partial \theta^3}/(6\frac{\partial^2 \mathrm{V}}{\partial \theta^2})$, $\beta_3 = -[\beta_1 + 2\beta_2(\pi - \theta_{eq})]/[3(\pi - \theta_{eq})^2]$ for Carter-Handy parameters. This formulation ensures that 3$^{\mathrm{rd}}$ derivatives($\frac{\partial^3 \mathrm{V}}{\partial S^3}$) are zero in the new coordinates, and for Carter-Handy coordinates, 1$^{\mathrm{st}}$ derivatives of CH with respect to Cartesians are zero when the bend angle is linear ($\frac{\partial(\mathrm{CH})}{\partial \mathbf{X}}\|_{\theta=\pi} = \mathbf{0}$). **Warning:** You need to set `DLVL_INT` $\geq$ `3`, and this procedure should only be used to estimate required parameters for a given molecule **at equilibrium**. Thereafter, the same parameters should be supplied explicitly for each coordinate.

Continued on next page

Table 3 – *Continued from previous page*

Command	**Options / Default / Description**
	This procedure has been found to work well for the Morse parameter, [2] but hasn't been tested for the Carter-Handy coordinate. In either case, it is important to double check that coordinates are sensible after transforming. Also, transformation between internals of the same type is allowed, e.g. if internal number 5 is Morse then `5/Morse/1.2` is valid, even if Morse parameter is already 1.2 bohr^{-1}.
`SAVE_FF_INT`	`True or False` Default: `False` Saves force-field as a Taylor series expansion, see Tute 3. Note that this requires an existing force field module named molecule.py or molecule_`MOLECULE`.py to be supplied in a subdirectory (named force_field) of the working directory. PyPES updates this file with the new force field. To reuse the created force field, the file should be manually renamed (to molecule_*newname*.py), and manually edited to ensure the PES number and force-field comments are appropriate, then copied to the code/force_field master directory. The force field is then accessible by referring to it as *newname*.
`SAVE_FF_NM`	`True or False` Default: `False` Saves derivatives of PES with respect to normal mode coordinates, see Tute 3. It is important to remember that derivatives unscaled by combinatorial factors are saved, so it is not a Taylor series expansion. Otherwise, all comments above for `SAVE_FF_INT` also apply here.
`EQUILIBRIUM`	`True or False` Default: `False`, except if using geometry stored in force-field module, then `True`. Determined automatically. When set to `True`, then when calculating $[\frac{\partial \mathbf{S}}{\partial \mathbf{X}}]$ and transforming **first derivatives** of PES, $\frac{\partial V}{\partial \mathbf{S}}$, to Cartesians, $\frac{\partial V}{\partial \mathbf{X}}$, derivative order is set to `DLVL_CART-1`. At equilibrium, the first derivative of PES is zero, so this setting ensures no time is wasted calculating higher order $[\frac{\partial \mathbf{S}}{\partial \mathbf{X}}]$ derivatives just to multiply them by zero. **Should not be set by user**

Table 4: Implemented force-fields. For definitions, refer to the original publications, or look in the force field module files (in `code/force_field` directory). When coordinate set is specified in curly brackets, the force-field was saved as Taylor series expansion after transforming derivatives to that coordinate set, or in case of normal mode coordinates, Q, derivatives themselves were saved. Atom ordering is indicated in brackets under each molecule name.

`MOLECULE`	`PESN`	**Description**
`H2O`	`1`	H_2O, original [3]
`(H,O,H)`	`2`	$\{f_M(r), \cos(\theta)\}$ from PESN 1
	`3`	$\{Q\}$ from PESN 2
`NH2`	`1`	NH_2^-, original [4]
`(H,N,H)`	`2`	$\{f_M(r), \cos(\theta)\}$ from PESN 1
	`3`	$\{Q\}$ from PESN 2
`HO2`	`1`	HO_2^+ (X^3A'') ground state, original [3]
`(H,O,O)`	`2`	HO_2^+ (A^1A') excited state, original [3]
	`3`	$\{f_M(r), \cos(\theta)\}$ from PESN 1
	`4`	$\{Q\}$ from PESN 3
	`5`	$\{f_M(r), \cos(\theta)\}$ from PESN 2
	`6`	$\{Q\}$ from PESN 5
`HOCl`	`1`	HOCl, original [5]
`(H,O,Cl)`	`2`	$\{Q\}$ from PESN 1
`HOBr`	`1`	HOBr, original [5]
`(H,O,Br)`	`2`	$\{Q\}$ from PESN 1

Continued on next page

Table 4 – *Continued from previous page*

MOLECULE	PESN	**Description**
HOF	1	HOF, original [6]
(H,O,F)	2	$\{f_M(r), \cos(\theta)\}$ from PESN 1
	3	$\{Q\}$ from PESN 2
PF2	1	PF_2^+ , original [7]
(F,P,F)	2	$\{Q\}$ from PESN 1
PO2	1	PO_2^- , original [7]
(O,P,O)	2	$\{Q\}$ from PESN 1
SO2	1	SO_2, original [7]
(O,S,O)	2	$\{Q\}$ from PESN 1
SiF2	1	SiF_2, original [7]
(F,Si,F)	2	$\{Q\}$ from PESN 1
F2O	1	F_2O , original [6]
(F,O,F)	2	$\{f_M(r), \cos(\theta)\}$ from PESN 1
	3	$\{Q\}$ from PESN 2
BrO2	1	BrO_2 (X^2B_1) ground state, original [8]
(O,Br,O)	2	BrO_2 (A^2A_2) excited state, original [8]
	3	$\{Q\}$ from PESN 1
	4	$\{Q\}$ from PESN 2
ClO2	1	ClO_2 (X^2B_1) ground state, original [8]
(O,Cl,O)	2	ClO_2 (A^2A_2) excited state, original [8]
	3	ClO_2^+ ion, original [7]
	4	$\{Q\}$ from PESN 1

Continued on next page

Table 4 – *Continued from previous page*

MOLECULE	PESN	**Description**
	5	$\{Q\}$ from PESN 2
	6	$\{Q\}$ from PESN 3
BiH3	1	BiH_3, original [9]
(Bi,H,H,H)	2	$\{Q\}$ from PESN 1
NF3	1	NF_3, original [10]
(N,F,F,F)	2	$\{f_M(r), \cos(\theta)\}$ from PESN 1
	3	$\{Q\}$ from PESN 2
NH3	1	NH_3, original [11]
(N,H,H,H)	2	$\{Q\}$ from PESN 1
PH3	1	PH_3, original [12]
(P,H,H,H)	2	$\{Q\}$ from PESN 1
SbH3	1	SbH_3, original [9]
(Sb,H,H,H)	2	$\{Q\}$ from PESN 1
SiH3	1	${SiH_3}^-$, original [13]
(Si,H,H,H)	2	$\{f_M(r), \cos(\theta)\}$ from PESN 1
	3	$\{Q\}$ from PESN 2
AlF3	1	AlF_3, original [14]
(Al,F,F,F)	2	$\{f_M(r), \cos(\theta), \sin(\omega_2)\}$ from PESN 1
	3	$\{Q\}$ from PESN 2
BF3	1	BF_3, original [15]
(B,F,F,F)	2	$\{f_M(r), \cos(\theta), \sin(\omega_2)\}$ from PESN 1
	3	$\{Q\}$ from PESN 2

Continued on next page

Table 4 – *Continued from previous page*

MOLECULE	PESN	Description
CF3	1	$CF_3{}^+$, original [15]
(C,F,F,F)	2	$\{f_M(r), \cos(\theta), \sin(\omega_2)\}$ from PESN 1
	3	$\{Q\}$ from PESN 2
SiF3	1	$SiF_3{}^+$, original [14]
(Si,F,F,F)	2	$\{f_M(r), \cos(\theta), \sin(\omega_2)\}$ from PESN 1
	3	$\{Q\}$ from PESN 2
SO3	1	SO_3, original [16]
(S,O,O,O)	2	$\{f_M(r), \cos(\theta), \sin(\omega_1)\}$ from PESN 1
	3	$\{Q\}$ from PESN 2
H2CO	1	H_2CO, original [17]
(O,C,H,H)	2	$\{Q\}$ from PESN 1
H2SiO	1	H_2SiO, original [18]
(H,H,Si,O)	2	$\{Q\}$ from PESN 1
N2H2	1	trans-N_2H_2, original [19]
(H,N,N,H)	2	$\{f_M(r), \cos(\theta), \sin(\tau)\}$ from PESN 1
	3	$\{Q\}$ from PESN 2
H2O2	1	H_2O_2, original [20]
(H,O,O,H)	2	$\{Q\}$ from PESN 1
	3	deuterated H_2O_2, $\{Q\}$ from PESN 1
HSOH	1	trans-HSOH, original [21]
(H,S,O,H)	2	$\{Q\}$ from PESN 1
HSiOH	1	cis-HSiOH, original [22]

Continued on next page

Table 4 – *Continued from previous page*

MOLECULE	PESN	**Description**
(H,Si,O,H)	2	trans-HSiOH, original [22]
	3	$\{f_M(r), \cos(\theta), \sin(\tau)\}$ from PESN 1
	4	$\{Q\}$ from PESN 3
	5	deuterated cis-HSiOH, $\{Q\}$ from PESN 3
	6	$\{f_M(r), \cos(\theta), \sin(\tau)\}$ from PESN 2
	7	$\{Q\}$ from PESN 6
	8	deuterated trans-HSiOH, $\{Q\}$ from PESN 6
HOCO	1	cis-HOCO, original [23]
(H,O,C,O)	2	trans-HOCO, original [24]
	3	$\{f_M(r), \cos(\theta), \sin(\tau)\}$ from PESN 1
	4	$\{Q\}$ from PESN 3
	5	deuterated cis-HOCO, $\{Q\}$ from PESN 3
	6	$\{f_M(r), \cos(\theta), \sin(\tau)\}$ from PESN 2
	7	$\{Q\}$ from PESN 6
	8	deuterated trans-HOCO, $\{Q\}$ from PESN 6
C4	1	C_4, original [25]
(C,C,C,C)	2	$\{f_M(r), \cos(\theta), \sin(\tau)\}$ from PESN 1
	3	$\{Q\}$ from PESN 2
CF4	1	CF_4, original [26]
(C,F,F,F,F)	2	$\{f_M(r), \cos(\theta)\}$ from PESN 1
	3	$\{Q\}$ from PESN 2
CH4	1	CH_4, original [27]
(C,H,H,H,H)	2	$\{f_M(r), \cos(\theta)\}$ from PESN 1

Continued on next page

Table 4 – *Continued from previous page*

MOLECULE	PESN	Description
	3	$\{Q\}$ from PESN 2
NH4	1	$NH_4{}^+$, original [28]
(N,H,H,H,H)	2	$\{f_M(r), \cos(\theta)\}$ from PESN 1
	3	$\{Q\}$ from PESN 2
SiF4	1	SiF_4, original [26]
(Si,F,F,F,F)	2	$\{f_M(r), \cos(\theta)\}$ from PESN 1
	3	$\{Q\}$ from PESN 2
SiH4	1	SiH_4, original [29]
(Si,H,H,H,H)	2	$\{f_M(r), \cos(\theta)\}$ from PESN 1
	3	$\{Q\}$ from PESN 2
SnH4	1	SnH_4, original [30]
(Sn,H,H,H,H)	2	$\{f_M(r), \cos(\theta)\}$ from PESN 1
	3	$\{Q\}$ from PESN 2
FClO3	1	$FClO_3$, original [31]
(Cl,F,O,O,O)	2	$\{f_M(r), \cos(\theta)\}$ from PESN 1
	3	$\{Q\}$ from PESN 2
OPH3	1	OPH_3, original [32]
(P,O,H,H,H)	2	$\{f_M(r), \cos(\theta)\}$ from PESN 1
	3	$\{Q\}$ from PESN 2
SPH3	1	SPH_3, original [32]
(P,S,H,H,H)	2	$\{f_M(r), \cos(\theta)\}$ from PESN 1
	3	$\{Q\}$ from PESN 2

Continued on next page

Table 4 – *Continued from previous page*

MOLECULE	PESN	**Description**
C3H2	1	C_3H_2, original [33]
(C,C,C,H,H)	2	$\{f_M(r), \theta, \tau\}$ from PESN 1
	3	$\{Q\}$ from PESN 2
C3H3	1	cyclic-$C_3H_3{}^+$, original [34]
(C,C,C,H,H,H)	2	$\{f_M(r), \cos(\theta), \sin(\omega_2)\}$ from PESN 1
	3	$\{Q\}$ from PESN 2
C2H4	1	C_2H_4, original [35]
(C,C,H,H,H,H)	2	$\{f_M(r), \theta, \tau\}$ from PESN 1
	3	$\{Q\}$ from PESN 2

4 Tutorials

A short summary of what's covered in each tutorial:

Tute 0 – minimal working examples, with all keywords except those pertaining to force field definition (`MOLECULE` and `PESN`) set by default;

Tute 1 – basic input specification, derivative evaluation at equilibrium, writing derivatives to separate files;

Tute 2 – geometry specification in Cartesian and internal coordinates, geometry optimisation w.r.t energy, mass specification, normal mode analysis;

Tute 3 – transformation between internal coordinates, saving force-field in a new set of internals or in normal mode coordinates;

Tute 4 – working with force-fields saved in normal mode coordinates;

Tute 5a/5b – using PyPES to generate benchmark Cartesian derivative data, analogous to *ab initio* packages, only 5a reproduced here. Also included in the 5a directory are example wrapper scripts that call PyPES and read in derivative data, for more information see the `README` file in the directory tute5a directory.

Tute 6 – structure of force-field modules, some comments on implementing new force-fields; not reproduced in user manual, for force field developers;

```
#============================================================
# Tutorial 0: - some minimal working examples
#============================================================
# Putting the hash symbol as the first character on a line will cause that line to be ignored
# @@@ symbol breaks up input file into multiple jobs
# Blank lines are ignored
#
# Job block 1: Prints force field info and energy to output file.
# If output file is not specified, you get nothing!
# COMMENTs not strictly required, but useful for readability

COMMENT        "Job block 1:  minimal example"
MOLECULE        AlF3

@@@

# Job block 2: Prints force field info, harmonic frequencies and normal mode
# coordinate vectors loaded from PES data file to output file
# For further examples on accessing stored NM deriv data, see tute 4

COMMENT         "Job block 2:  minimal example with normal mode analysis"
MOLECULE        AlF3
PESN            3
```

```
#============================================================
# Tutorial 1: - basic input specification
# - derivative evaluation at equilibrium
# - special characters
#============================================================

# Putting a hash symbol as a first character on a line will cause that line to be ignored.
# Empty lines are ignored as well. When reading commands they will be checked for typos
# and an error will be raised if they are invalid.
# Commands can be specified in any order.
# Multiple jobs per input file are allowed, separated by @@@.
# Commands are not case sensitive.

# Lets have a look at some examples.

COMMENT          "Job block 1"
MOLECULE         SO2
PESN             1
DLVL_INT         6
# DLVL_CART      6    this is set by default
PRINT_DERIV_INT  True
PRINT_DERIV_CART True
PRINT_DERIV_NM   True
# Cartesian geometry is set to equilibrium by default
@@@
# For my work, it was convenient to run dozens, sometimes over a hundred, jobs from the same
# input file. So I've implemented horizontal propagation of commands within a block.
# Horizontal propagation is initiated by specifying multiple MOLECULE options.
# Options to other commands pertain to each specified molecule according position
# If only one option is specified, it will be assumed to pertain to all molecules
```

```
# Any options that exceed the number of MOLECULE options are not read.
COMMENT          "Job block 2:  example of horizontal propagation of jobs"
MOLECULE         SO2     H2O     HO2
PESN             1       2       3
DLVL_INT         6       4       4      not_read
DLVL_CART        6
PRINT_DERIV_INT  True
PRINT_DERIV_CART True
PRINT_DERIV_NM   True
# NOTE: horizontal propagation does not work with CARTESIAN, INTERNAL and MASSES commands.
# Use COORDS_FILE_CART, COORDS_FILE_INT and MASSES_FILE instead
@@@
# semicolon, ";", is a special character that causes current line to be wrapped.
# It is most convenient with COMMENT, however it works the same with any input line.
# I would advise you to use it only in COMMENT section, though.
COMMENT         "Job block 3:;
example of semicolon use"
MOLECULE        N2H2    HSiOH ;
H2SiO
PESN            2       2   ;
1
DLVL_INT;
4       4       4
DLVL_CART       6
PRINT_DERIV_INT  True
PRINT_DERIV_CART True
PRINT_DERIV_NM   True
# There is usually no reason to use it, except with COMMENT.
@@@
# Here is an example for writing derivatives to separate files.
```

```
# Wrappers for reading those derivatives into arrays for different languages are included.
COMMENT           "Job block 4:  printing derivatives to separate files"
MOLECULE          SPH3         C2H4
PESN              2            3
DLVL_INT          4
PRINT_DERIV_INT   True
PRINT_DERIV_CART  True
PRINT_DERIV_NM    True
PRINT_DERIV_FILE  default     ethylene
STOP # this is a last special "character". It prevents the rest of the file from being read/processed.
```

```
#============================================================
```

Tutorial 2: - specifying masses

- running normal mode analysis

- specifying geometry in Cartesian coordinates

- specifying geometry in internal coordinates

- geometry optimisation w.r.t energy

```
#===========================================================

COMMENT          "Job block 1:  changing masses and calling normal mode"
MOLECULE         SO2
PESN             1
DLVL_INT         2
DLVL_CART        2
PRINT_DERIV_INT  True
PRINT_DERIV_CART True
DO_NORMAL_MODE   True
```

there is usually no need to explicitly invoke `DO_NORMAL_MODE`,

when you ask derivatives in NM coordinates to be printed

or to save them in `force_field` module, it is done by default

```
MASSES
1 O18
3 17.9991610
END
```

Oxygen atom 1 is assigned the mass of ^{18}O using the built-in isotope dictionary

Oxygen atom 3 is assigned the mass of ^{18}O manually

```
@@@
```

There are two ways of specifying a geometry.

The first is by giving Cartesian coordinates directly in the input file

```
COMMENT          "Job block 2:  use of CARTESIAN"
MOLECULE         SO2
```

```
PESN             1
DLVL_INT         2
DLVL_CART        2
PRINT_DERIV_INT  True
PRINT_DERIV_CART True
```

Another way of changing masses is by specifying new atom types (actually isotopes) during

geometry specification in `CARTESIAN`, although an explicit `MASSES` section would override this.

In this format horizontal propagation of command options cannot be used

```
CARTESIAN bohr
O18      2.35     0.69    0.0
S        0.00    -0.69    0.0
O18     -2.34     0.69    0.0
@@@
COMMENT          "Job block 2:  use of COORDS_CART_FILE, MASSES_FILE"
MOLECULE         SO2              CF4
PESN             1                2
DLVL_INT         2
DLVL_CART        2
PRINT_DERIV_INT  True
PRINT_DERIV_CART True
DO_NORMAL_MODE   True
MASSES_FILE      masses_SO2.m     masses_CF4.m
```

this can also be used to specify masses.

```
COORDS_CART_FILE geom_SO2.cart    geom_CF4.cart
```

same job as before, followed by an analogous calculation on an isotopically substituted CF4.

Note that `PRINT_DERIV_NM` will do normal mode analysis to get normal mode derivatives,

so it can't be used off-equilibrium unless you specify normal mode coordinates separately

Look at Tute 3 to see how it's done.

```
@@@
```

Another way of specifying a geometry is by giving it in internal coordinates.

```
COMMENT          "Job block 3:  use of INTERNAL"
MOLECULE         SO2
PESN             1
DLVL_INT         2
DLVL_CART        2
PRINT_DERIV_INT  True
PRINT_DERIV_CART True
INTERNAL bohr
1   BL           3.0
2   BL           2.0
3   BA         100.0
END
```

in the `INTERNAL` block only the first and last column are read. It is not necessary

to specify internal types, but it is convenient.

Internals must be specified using BL, BA, DA, OOP1, or OOP2 types,

and all angular coordinates must be given in degrees.

So, even if the force-field is saved in {Morse,CosBA,SinDA} coordinate system,

the geometry must be specified in {BL,BA,DA} coordinate system

You don't have to use all of the internals to constrain the geometry, which is why `END` is necessary.

`@@@`

The INTERNAL block can also read from file.

If you are only interested in obtaining the least-squares fitted Cartesian coordinates,

setting `DLVL_INT` to -1 will exit before derivative calculations

```
COMMENT         "Job block 4:  use of COORDS_INT_FILE"
MOLECULE        SO2
PESN            1
DLVL_INT        -1
COORDS_CART_FILE geom_SO2.cart
COORDS_INT_FILE  geom_SO2.int
```

When CARTESIAN is specified concurrently, then that Cartesian geometry is used as a
starting point for optimisation of Cartesian coordinates to satisfy specified internal values
```
@@@
```
Lastly, if you don't have an optimized geometry, you can use `scipy.minimize` to find it

```
COMMENT          "Job block 5:  use of OPT_GEOM_WRT_E"
MOLECULE         SO2
PESN             1
DLVL_INT         2
DLVL_CART        2
PRINT_DERIV_INT  True
PRINT_DERIV_CART True
PRINT_DERIV_NM   True
COORDS_CART_FILE geom_SO2.cart
OPT_GEOM_WRT_E   True
```

NOTE: you can play around with minimize() settings if you have convergence issues;
have a look in `read_input.py`

```
#==================================================================
# Tutorial 3: - transforming PES derivatives between internals
# - specifying normal mode coordinates
# - saving ff in internal coordinates
# - saving ff in normal mode coordinates
#==================================================================

# I've done quite a bit of writing on coordinate transformation between internals,
# so I think a few examples will suffice on that topic.
# I would like to reiterate though, that you should double check that the new
# coordinate behaves sensibly. Because they do have a range of validity, beyond which
# they can become unstable, undefined, or behave unphysically.
COMMENT          "Job block 1:  options 1a) and 1b) to NEW_INT_TYPE"
MOLECULE         SO2
PESN             1
DLVL_INT         6
NEW_INT_TYPE     1/SPF
NEW_INT_TYPE     2/SPF
NEW_INT_TYPE     3/CH/1.0/-0.23/-0.15
DLVL_CART        6
PRINT_DERIV_INT  True
PRINT_DERIV_CART True
PRINT_DERIV_NM   True
@@@
# Transformations between different internal coordinates can also be done off-equilibrium.
COMMENT          "Job block 2:  option 2)"
MOLECULE         SO2     SO2
PESN             1
DLVL_INT         6
NEW_INT_TYPE     BL/SPF  BL/Morse/1.0
```

```
NEW_INT_TYPE      CH/BA  CH/CosBA
DLVL_CART         5
NEW_NM_COORDS     coords_SO2.nm
PRINT_DERIV_INT   True
PRINT_DERIV_CART  True
PRINT_DERIV_NM    True
COORDS_CART_FILE  geom_SO2.cart
# But if you want to print out derivatives in NM coordinates then you
# need to specify them manually, since doing normal mode analysis off-equilibrium
# will give you the wrong coordinates. See coord_SO2.nm for the format
@@@
# When saving ff you need a directory called force_field in the same directory, within that
# you need an example module; molecule_MOLECULE.py or molecule.py
# These can be copied from the originals supplied in code/force_field
# If you want to work with the newly generated force field in future you will need to copy it
# into code/force_field, changing MOLECULE to something else, which will become the new
# name by which the force field can be invoked using the MOLECULE keyword.
COMMENT           "Job block 3:  Saving ff in internals and NM coordinates"
MOLECULE          H2O
PESN              1
DLVL_INT          4
NEW_INT_TYPE      BL/Morse
NEW_INT_TYPE      BA/CosBA
DLVL_CART         6
PRINT_DERIV_INT   True
PRINT_DERIV_CART  True
PRINT_DERIV_NM    True
SAVE_FF_INT       True
SAVE_FF_NM        True
# Afterwards you will need to have a look at the saved ff and check the PES numbering,
```

```
# write a sensible force field comment, and uncomment the else statement if you started
# with molecule.py
# Note that with SAVE_FF_NM only derivatives are saved without any scaling factors.
```

#===

Tutorial 4: - some points on force fields saved in normal mode coordinates

#===

Most people will only want derivatives of the PES with respect to normal mode coordinates
from this library, and it seemed like a waste of time for everyone to have to go through
all the differentiation and coordinate transformation steps to obtain those derivatives.
Therefore, they have been saved separately without scaling factors *i.e.* not in Taylor
series expansion form.

Important note: only the derivatives of the energy with respect to the 3N-6 vibrational
normal modes are saved. Therefore, while it is possible to back-transform from normal mode
derivatives into Cartesian and internal coordinates, it is not recommended, as the neglect of
translational and rotational modes means that these derivative sets will not be the same as
obtained from the original PES defined in terms of internal coordinates.

A simple example is H_2O, in which the vibrational normal modes are all within the plane
of the molecule. But at higher orders of expansion of the PES (beyond 2^{nd} order), motion out
of the plane distorts the geometry leading to non-zero Cartesian derivatives which can only
be described using rotational normal modes.

Consequently, it is **not recommended** to use the stored normal mode derivatives as
a basis for subsequent coordinate transformation steps, except perhaps rotating normal modes.
(see job block 2 below). Even so, if in doubt, just start with the force field in internal
coordinates, as the time difference will not be large.

`DLVL_INT` is the only relevant derivative level specifier, used for both normal mode and Cartesian
derivatives of PES

```
COMMENT          "Job block 1:  print stored NM coords and approx Cartesian derivs"
MOLECULE         C3H2
PESN             3
DLVL_INT         4
PRINT_DERIV_CART True
PRINT_DERIV_NM   True
@@@
```

Changing to a new NM basis

```
COMMENT          "Job block 2:  changing coordinates"
MOLECULE         C3H2
PESN             3
DLVL_INT         4
PRINT_DERIV_CART True
PRINT_DERIV_NM   True
NEW_NM_COORDS    coords_C3H2.nm
```

```
#=============================================================================
# Tutorial 5a: - using PyPES to replicate functionality of ab initio program
#=============================================================================

# To evaluate the energy and its derivatives with respect to Cartesian coordinates
# at a range of different geometries, it is only necessary to set up the PyPES input
# file once, and update the geometry through COORDS_CART_FILE.
# In this example, the base file name for Cartesian derivative data is set manually
# Note that specifying an output file is optional, as the wrapper scripts are
# set up to read in the derivative data output to file

COMMENT           "Job block 1:  4th derivs of energy w.r.t Cartesian coordinates"
MOLECULE          HOCO
PESN              1       # cis-HOCO
DLVL_INT          4
PRINT_DERIV_CART  True
PRINT_DERIV_FILE  cis_HOCO_qff
COORDS_CART_FILE  coords_cis_HOCO.cart
```

References

1. Sibaev, M.; Crittenden, D. L. *submitted.*

2. Fortenberry, R. C.; Huang, X.; Yachmenev, A.; Thiel, W.; Lee, T. J. *Chem. Phys. Lett.* **2013**, *574*, 1–12.

3. Huang, X.; Lee, T. J. *J. Chem. Phys.* **2008**, *129*(4), 044312–044312–14.

4. Huang, X.; Lee, T. J. *J. Chem. Phys.* **2009**, *131*(10), 104301–104301–15.

5. Peterson, K. A. *Spectrochim. Acta Part A: Mol. Biomol. Spectrosc.* **1997**, *53*(8), 1051–1064.

6. Breidung, J.; Thiel, W.; Gauss, J.; Stanton, J. F. *J. Chem. Phys.* **1999**, *110*(8), 3687.

7. Pak, Y.; Woods, R. C. *J. Chem. Phys.* **1996**, *104*(14), 5547.

8. Peterson, K. A. *J. Chem. Phys.* **1998**, *109*(20), 8864.

9. Yurchenko, S. N.; Thiel, W.; Jensen, P. *J. Mol. Spec.* **2006**, *240*(2), 174–187.

10. Tarroni, R.; Palmieri, P.; Senent, M. L.; Willetts, A. *Chem. Phys. Lett.* **1996**, *257*(1), 23–30.

11. Yurchenko, S. N.; Zheng, J.; Lin, H.; Jensen, P.; Thiel, W. *J. Chem. Phys.* **2005**, *123*(13), 134308–134308–14.

12. Ovsyannikov, R. I.; Thiel, W.; Yurchenko, S. N.; Carvajal, M.; Jensen, P. *J. Chem. Phys.* **2008**, *129*(4), 044309–044309–8.

13. Aarset, K.; Császár, A. G.; Sibert III, E. L.; Allen, W. D.; Schaefer III, H. F.; Klopper, W.; Noga, J. *J. Chem. Phys.* **2000**, *112*(9), 4053–4063.

14. Pak, Y.; Sibert, E.; Woods, R. *J. Chem. Phys.* **1997**, *107*(6), 1717–1724.

15. Pak, Y.; Woods, R. C. *J. Chem. Phys.* **1997**, *106*(15), 6424.

16. Martin, J. *Spectrochim. Acta Part A: Mol. Biomol. Spectrosc.* **1999**, *55*(3), 709–718.

17. Yachmenev, A.; Yurchenko, S. N.; Jensen, P.; Thiel, W. *J. Chem. Phys.* **2011**, *134*(24), 244307–244307–11.

18. Koput, J.; Carter, S.; Handy, N. C. *Chem. Phys. Lett.* **1999**, *301*(1), 1–9.

19. Martin, J. M. L.; Taylor, P. R. *Spectrochim. Acta Part A: Mol. Biomol. Spectrosc.* **1997**, *53*(8), 1039–1050.

20. Malyszek, P.; Koput, J. *J. Comput. Chem.* **2013**, *34*(5), 337–345.

21. Yurchenko, S. N.; Yachmenev, A.; Thiel, W.; Baum, O.; Giesen, T. F.; Melnikov, V. V.; Jensen, P. *J. Mol. Spec.* **2009**, *257*(1), 57–65.

22. Martin, J. *J. Phys. Chem. A* **1998**, *102*(8), 1394–1404.

23. Fortenberry, R. C.; Huang, X.; Francisco, J. S.; Crawford, T. D.; Lee, T. J. *J. Chem. Phys.* **2011**, *135*(21), 214303–214303–10.

24. Fortenberry, R. C.; Huang, X.; Francisco, J. S.; Crawford, T. D.; Lee, T. J. *J. Chem. Phys.* **2011**, *135*(13), 134301–134301–8.

25. Wang, X.; Huang, X.; Bowman, J. M.; Lee, T. J. *J. Chem. Phys.* **2013**, *139*(22), 224302.

26. Wang, X.-G.; Sibert, E. L.; Martin, J. M. L. *J. Chem. Phys.* **2000**, *112*(3), 1353–1366.

27. Lee, T. J.; Martin, J. M. L.; Taylor, P. R. *J. Chem. Phys.* **1995**, *102*(1), 254–261.

28. Martin, J.; Lee, T. *Chem. Phys. Lett.* **1996**, *258*(1-2), 129–135.

29. Martin, J.; Baldridge, K.; Lee, T. *Mol. Phys.* **1999**, *97*(8), 945–953.

30. Fusina, L.; Nivellini, G.; Salzillo, T.; Lamarra, M.; Tarroni, R. *J. Chem. Phys.* **2012**, *137*(20), 204316–204316–10.

31. Cane, E.; Fusina, L.; Lamarra, M.; Tarroni, R.; Burczyk, K. *J. Phys. Chem. A* **2008**, *112*(51), 13729–13736.

32. Lamarra, M.; Tarroni, R. *Mol. Phys.* **2011**, *109*(17-18), 2095–2104.

33. Lee, T. J.; Huang, X.; Dateo, C. E. *Mol. Phys.* **2009**, *107*(8-12), 1139–1152.

34. Huang, X.; Taylor, P. R.; Lee, T. J. *J. Phys. Chem. A* **2011**, *115*(19), 5005–5016.

35. Delahaye, T.; Nikitin, A.; Rey, M.; Szalay, P.; Tyuterev, V. *J. Chem. Phys.* **2014**, *141*(10), 104301.